Access Control based on Privileges

Discretionary access control in a database system is based on the granting and revoking of privileges.

In some cases it is desirable to grant a privilege to a user temporarily. For example, the owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed. Hence, a mechanism for revoking privileges is needed. In SQL a REVOKE command is included for the purpose of canceling privileges.

Whenever the owner A of a relation R grants a privilege on R to another account B, the privilege can be given to B with or without the GRANT OPTION. If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts. Suppose that B is given the GRANT OPTION by A and that B then grants the privilege on R to a third account C, also with the GRANT OPTION. In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R. If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.

It is possible for a user to receive a certain privilege from two or more sources. For example, A4 may receive a certain UPDATE R privilege from both A2 and A3. In such a case, if A2 revokes this privilege from A4, A4 will still continue to have the privilege by virtue of having been granted it from A3. If A3 later revokes the privilege from A4, A4 totally loses the privilege. Hence, a DBMS that allows propagation of privi-leges must keep track of how all the privileges were granted so that revoking of priv-ileges can be done correctly and completely.

Role-Based Access Control

Role-based access control (RBAC) emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprise-wide systems. Its basic notion is that privileges and other permissions are associated with organizational **roles**, rather than individual users. Individual users are then assigned to appropriate roles. Roles can be created using the CREATE ROLE and DESTROY ROLE commands. The GRANT and REVOKE commands can be used to assign and revoke privileges from roles, as well as for individual users when needed. For example, a company may have roles such as

sales account manager, purchasing agent, mailroom clerk, department manager, and so on. Multiple individuals can be assigned to each role. Security privileges that are common to a role are granted to the role name, and any individual assigned to this role would automatically have those privileges granted.

RBAC can be used with traditional discretionary and mandatory access controls; it ensures that only authorized users in their specified roles are given access to certain data or resources. Users create sessions during which they may activate a subset of roles to which they belong. Each session can be assigned to several roles, but it maps to one user or a single subject only. Many DBMSs have allowed the concept of roles, where privileges can be assigned to roles.

Separation of duties is another important requirement in various commercial DBMSs. It is needed to prevent one user from doing work that requires the involvement of two or more people, thus preventing collusion. One method in which sepa-ration of duties can be successfully implemented is with mutual exclusion of roles. Two roles are said to be **mutually exclusive** if both the roles cannot be used simultaneously by the user. **Mutual exclusion of roles** can be categorized into two types, namely *authorization time exclusion (static)* and *runtime exclusion (dynamic)*. In authorization time exclusion, two roles that have been specified as mutually exclusive cannot be part of a user's authorization at the same time. In runtime exclusion, both these roles can be authorized to one user but cannot be activated by the user at the same time. Another variation in mutual exclusion of roles is that of complete and partial exclusion.

The **role hierarchy** in RBAC is a natural way to organize roles to reflect the organization's lines of authority and responsibility. By convention, junior roles at the bottom are connected to progressively senior roles as one moves up the hierarchy. The hierarchic diagrams are partial orders, so they are reflexive, transitive, and antisymmetric. In other words, if a user has one role, the user automatically has roles lower in the hierarchy. Defining a role hierarchy involves choosing the type of hierarchy and the roles, and then implementing the hierarchy by granting roles to other roles. Role hierarchy can be implemented in the following manner:

```
GRANT ROLE full_time TO employee_type1

GRANT ROLE intern TO employee_type2
```

The above are examples of granting the roles *full_time* and *intern* to two types of employees.

Another issue related to security is *identity management*. **Identity** refers to a unique name of an individual person. Since the legal names of persons are not necessarily unique, the identity of a person must include sufficient additional information to make the complete name unique. Authorizing this identity and managing the schema of these identities is called **Identity Management.** Identity Management addresses how organizations can effectively authenticate people and manage their access to confidential information. It has become more visible as a business requirement across all industries affecting organizations of all sizes. Identity Management administrators constantly need to satisfy application owners while keeping expenditures under control and increasing IT efficiency.

Another important consideration in RBAC systems is the possible temporal constraints that may exist on roles, such as the time and duration of role activations, and timed triggering of a role by an activation of another role. Using an RBAC model is a highly desirable goal for addressing the key security requirements of Web-based applications. Roles can be assigned to workflow tasks so that a user with any of the roles related to a task may be authorized to execute it and may play a certain role only for a certain duration.

RBAC models have several desirable features, such as flexibility, policy neutrality, better support for security management and administration, and other aspects that make them attractive candidates for developing secure Web-based applications. These features are lacking in DAC and MAC models. In addition, RBAC models include the capabilities available in traditional DAC and MAC policies. Furthermore, an RBAC model provides mechanisms for addressing the security issues related to the execution of tasks and workflows, and for specifying user-defined and organization-specific policies. Easier deployment over the Internet has been another reason for the success of RBAC models.